Efficient Mining of Discriminative Itemsets

Majid Seyfi
Queensland University of Technology
Science and Engineering Faculty
Brisbane, Australia
majid.seyfi@hdr.qut.edu.au

Richi Nayak
Queensland University of Technology
Science and Engineering Faculty
Brisbane, Australia
r.nayak@qut.edu.au

Yue Xu
Queensland University of Technology
Science and Engineering Faculty
Brisbane, Australia
yue.xu@qut.edu.au

Shlomo Geva
Queensland University of Technology
Science and Engineering Faculty
Brisbane, Australia
s.geva@qut.edu.au

ABSTRACT

Discriminative itemsets can be more useful than frequent itemsets as the former identifies the frequent itemsets in one dataset with much higher frequencies than the same itemsets in other datasets. The discriminative itemsets can distinguish the target dataset from all others. The discriminative itemsets are a small subset of frequent itemsets. The efficient mining of discriminative itemsets is a challenging problem, since the *Apriori* property of frequent itemsets is not applicable, and the designed algorithms must deal with the exponential number of itemset combinations in more than one dataset. In this paper, a novel algorithm, called *DISSparse,* is proposed for efficient mining of discriminative itemsets. Two determinative heuristics are proposed for limiting the mining of discriminative itemsets to the potential discriminative itemsets. Our experiments show the efficient time and space usage of the proposed algorithm in the large and complex datasets.

CCS CONCEPTS

• **Information systems→ Data mining** • **Information systems→ Association rules**

KEYWORDS

Data mining; Discriminative itemsets; Prefix-tree

ACM Reference format:

M. Seyfi, R. Nayak, Y. Xu and S. Geva. 2017. Efficient Mining of Discriminative Itemsets. In *Proceedings of WI '17, Leipzig, Germany, August 23-26, 2017,* 8 pages. http://dx.doi.org/10.1145/3106426.3106429

1 INTRODUCTION

The discriminative itemset mining across multiple datasets captures the itemsets which are highly frequent in one dataset, but less frequent or infrequent in other datasets. For simplicity, we call other datasets a 'general dataset'. The discriminative itemsets are relatively frequent in the target dataset and relatively infrequent in the general dataset.

There are many real world scenarios that can show the significance of discriminative itemset mining. In monitoring of market basket transactions, the itemsets that occur more frequently in one market compared to the other markets are of interest. These are useful for identifying the specific set of items which are of high interest in one market compared to the other markets. In an application of web page personalization, the discriminative itemsets are groups of the web pages visited together by specific user groups much more frequently than other user groups. In search engines, the discriminative itemsets are the sequences of queries asked more frequently in one geographical area compared to another area. In network traffic monitoring, discriminative itemsets are the concurrent activities of one user, which are more frequent in comparison to the rest of the same group activities in the network. Discriminative itemsets highlight the differences between datasets [1, 2]. They can be used for classification methods by distinguishing the trends in the target dataset from other datasets.

Contrast mining is a focused data mining research area for discovering interesting contrast patterns that state the significant differences between datasets [3]. The emerging patterns [4] are one of the well-known contrast patterns. In the emerging patterns the degree of change in supports of itemsets is important, and the actual support of itemsets is not considered.

In contrast, the discriminative itemsets proposed in this paper focus on the difference in support rather than the change degree of support. We discover the real support of each discriminative itemset and the relative differences of supports in datasets explicitly, which can provide concrete information assisting users in making right decisions. The δ-discriminative emerging patterns are special type of useful emerging patterns [5] which are frequent in the target dataset with frequency less than δ in the other datasets. The discriminative itemsets proposed in this paper are determined based on their relative occurrences in the target and general dataset. If the support of a pattern is relatively different in the target dataset and general dataset the pattern is considered discriminative (e.g., the itemsets in market basket frequent in all suburbs with relatively higher frequency in the target suburb). The δ-discriminative emerging patterns are determined based on their frequency (i.e., $< \delta$) in the general dataset.

The major challenge is that the problem of mining discriminative itemsets does not follow the *Apriori* property defined for the frequent itemset mining and a subset of discriminative itemsets can be non-discriminative. Additionally, any discriminative itemset mining method has to deal with the exponential number of itemsets generated in more than one dataset. The discriminative itemsets are usually a small subset of frequent itemsets. Finding a small set of discriminative itemsets from an exponential number of itemsets is time-consuming. In this paper, we present a novel and efficient algorithm, called *DISSparse*, based on the sparsity characteristics of discriminative itemsets. To deal with the itemset combination explosion, two heuristics are proposed to restrict the mining discriminative itemsets to the potential itemsets. The proposed heuristics are used for pruning the non-potential discriminative itemsets, which are infrequent in the first dataset or have growth rate less than defined threshold between datasets. It can be proved that the *DISSparse* method has full accuracy and recall in mining discriminative itemsets in one batch of transactions. We conduct extensive experiments; the results show that the *DISSparse* can deal with large and complex datasets. In summary, we make the following contributions:

- Developing the efficient *DISSparse* algorithm for mining discriminative itemsets;
- Proposing two heuristics for eliminating impossible discriminative itemsets to increase the efficiency;
- Introducing novel in-memory data structures for efficiently generating the potential discriminative itemsets;

The rest of the paper is organized as follows: Section 2 provides a review to the related works. In Section 3 we formally define the problem. Section 4 describes the previously proposed *DISTree* method. Section 5 presents the new *DISSparse* method in detail. Performance evaluation of the proposed method is presented in Section 6, and Section 7 offers the conclusion.

2 RELATED WORKS

One of the related works is Emerging Pattern mining (*EP*) [4]. *EPs* are itemsets whose growth rates are significantly higher in one dataset in comparison to another one. *EPs* are able to highlight the emerging trends in the time stamped datasets and also show the differences between datasets. The proposed method in [4], which has been followed in [6-12], works based on borders of maximal itemsets in the first and second dataset. The maximal itemsets are extracted for each dataset separately based on the defined growth rate threshold. The maximal itemsets are then reported altogether between the two borders defined using the lowest and the highest thresholds [4], as the Left and Right borders. By working on these two borders it limits the answers in the area between <Left, Right> to show a large collection of itemsets. There are differences between discriminative itemset mining and *EP* mining in terms of concepts and algorithms. Firstly, in *EPs* the degree of change in supports of itemsets is important, and the actual support of itemsets is not considered [4]. *EPs* can be infrequent, which could result in many *EPs* with low supports. In contrast, discriminative itemsets have to be frequent in the target dataset based on the support threshold, and be discriminative in the target dataset compared to the general dataset based on discriminative level. Secondly, the proposed algorithms for emerging patterns are mostly focused on representing these patterns in a compact way to avoid examining all the possible itemsets. The real supports of the *EPs* are not explicitly presented as they are reported in a group between two borders using the maximal itemsets. In the discriminative itemsets proposed in this paper, the frequencies of discriminative itemsets are derived and explicitly provided to the user together with the patterns.

In [1] three different methods are proposed for mining discriminative items in data streams, namely, frequent item based, hash-based method and hybrid method. These methods are considered as the first research works on mining discriminative items in data streams. The hierarchical counters approach [13] is proposed for identifying the exact frequencies of all the items including the infrequent items in the datasets and then discovering the discriminative items. These methods only find discriminative items which are not satisfactory for many real applications.

The δ-discriminative emerging patterns proposed in [5] are determined based on a threshold δ. The algorithm *DPMiner* in [5] can efficiently mine the δ-discriminative emerging patterns by skipping the subset of itemsets if their support in the general dataset is larger than δ. However, for the discriminative itemsets proposed in this paper, a subset of a non-discriminative itemsets can be discriminative and we cannot set a limit for the itemset frequency in the general dataset. The δ-discriminative emerging patterns must be infrequent in the general dataset which could exclude some of useful discriminative patterns compared to the discriminative itemsets proposed in this paper (e.g., the discriminative itemsets which are frequent in both the target dataset and general dataset, but the frequency is relatively different in the two datasets).

The most related work to this paper is the *DISTree* method proposed in [2], which is an adaptation of the *FP-Growth* method [14] for multiple datasets, can generate discriminative itemsets.

However, this method is inefficient and cannot deal with large datasets.

3 PROBLEM STATEMENT

Let $\sum$ be the alphabet set of items, a transaction $T = \{e_1, \dots e_i, e_{i+1}, \dots, e_n\}$, $e_i \in \sum$, is defined as a set of items in $\sum$. The two datasets S_i and S_j are defined as the target and general dataset; each consists of a different number of transactions, i.e., n_i and n_j, respectively. An itemset I is defined as a subset of $\sum$. The itemset frequency (often called absolute support) is the number of transactions that contain the itemset. The frequency of the itemset I in dataset S_i is denoted as $f_i(I)$ and the frequency ratio (often called relative support) of itemset I in S_i is defined as $r_i(I) = f_i(I)/n_i$. In this paper, if the frequency ratio of itemset I in the target dataset S_i is larger than the frequency ratio in the general dataset S_j, i.e., $\frac{r_i(I)}{r_j(I)} > 1$, then the itemset I can be considered as a discriminative itemset. Let $R_{ij}(I)$ be the discriminative value between $r_i(I)$ and $r_j(I)$, i.e., $R_{ij}(I) = \frac{r_i(I)}{r_j(I)}$. Obviously, the higher the $R_{ij}(I)$, the more discriminative the itemset I is.

To more accurately define discriminative itemsets, we introduce a user-defined threshold $\theta > 1$, called a discriminative level threshold with no upper bound. An itemset I is considered discriminative if $R_{ij}(I) \geq \theta$. This is formally defined as:

$$R_{ij}(I) = \frac{r_i(I)}{r_j(I)} = \frac{f_i(I)n_j}{f_j(I)n_i} \geq \theta \quad (1)$$

In the case of very small $f_j(I)$ or $f_j(I) = 0$, the $R_{ij}(I)$ could be very large but with very low $f_i(I)$. In order to accurately identify discriminative itemsets which have reasonable frequency, especially in the case of $f_j(I) = 0$, we introduce another user-specified support threshold, $0 < \varphi < 1/\theta$, to eliminate itemsets that have very low frequency. In this paper, an itemset I is considered as discriminative if its frequency becomes greater than $\varphi\theta n_i$ i.e., $f_i(I) \geq \varphi\theta n_i$.

Definition 1. (*Discriminative itemsets*) Let S_i and S_j be two datasets, with the size of n_i and n_j respectively, that contain varied length transactions of items in $\sum$, a user defined discriminative level threshold $\theta > 1$ and a support threshold $\varphi \in (0, 1/\theta)$. A set of discriminative itemsets in S_i against S_j, denoted as DI_{ij}, is formally defined as:

$$DI_{ij} = \{I \subseteq \sum \mid f_i(I) \geq \varphi\theta n_i \,\&\, R_{ij}(I) \geq \theta\} \quad (2)$$

The itemsets that are not discriminative are defined as non-discriminative itemsets.

4 DISTREE METHOD

The *DISTree* method proposed in [2] by simply expanding the *FP-Growth* method [14] is inefficient for finding discriminative itemsets in large datasets. The algorithm was designed based on several data structures either modified from the standard *FP-Growth* method of frequent itemset mining, or specifically developed for discriminative itemset mining as below.

FP-Tree: The prefix tree structure proposed in the *FP-Growth* [14] is used for holding the frequent items of the dataset by sharing the branches for their most common frequent items. The *FP-Tree* is adapted by adding two counters in each node for holding the frequencies of the itemsets in the target dataset and the general dataset (e.g., there are two counters associated with each node in the *FP-Tree* in Figure 1).

Header-Table: The *Header-Table* is a tabular structure showing all the items in the dataset by considering the processing order from the least frequent items. For fast traversing the prefix tree structures, each item is associated with two linked-lists, which hold the itemsets ending with that item in *FP-Tree* and *DISTree*, respectively (e.g., the *Header-Table* has links to the *FP-Tree* and *DISTree* in Figure 1 and Figure 2).

DISTree: The *DISTree* is a similar prefix tree structure to *FP-Tree* with two counters in each node. The two counters f_i and f_j show the frequencies of the itemsets in the sequence ending with each particular node in the target dataset and the general dataset, respectively. The *DISTree* is constructed by traversing through the links in the *Header-Table* structure following the *FP-Growth* method [14], generating all the combinations of itemsets that are frequent in the target dataset and testing their discriminative level in order to determine discriminative itemsets. The non-discriminative itemsets generated during the process are not included in *DISTree* unless they are a subset of discriminative itemsets (e.g., $c_{12,13}$ in Figure 2 is the non-discriminative itemset staying as the internal node).

4.1 DISTree Algorithm

The *FP-Tree* is constructed first for the transactions in both datasets. For every new transaction in S_i either a new prefix or a sub-prefix is added to the *FP-Tree* and the frequency pairs are added or the frequency pairs are updated in the prefixes in *FP-Tree* if the itemset already appeared in the past transactions. The process is continued by generating the itemsets which are frequent in the target dataset using *FP-Growth* [14] and adding them to the *DISTree* structure. The non-discriminative itemsets are ignored for space saving if they are not a subset of any discriminative itemsets.

Example 1. The *DISTree* construction is graphically monitored using the running example presented in Table 1 which contains two simple datasets with the same number of transactions in S_1 and S_2 ($n_1 = n_2 = 15$).

Table 1: A small input batch of transactions

S/T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	abcd	abcd	abcd	abc	ab	ace	bce	bc	bc	bde	bd	cde	cde	cde	ce
S_2	abcd	abcd	ac	ac	ac	ac	a	a	bcd	cde	cde	cde	cd	c	c

In this example the discriminative level threshold is set to $\theta = 2$ and the support threshold is set to $\varphi = 0.1$. The *FP-Tree*

and *DISTree* within *Header-Table* structure is represented in Figure 1 and Figure 2, respectively.

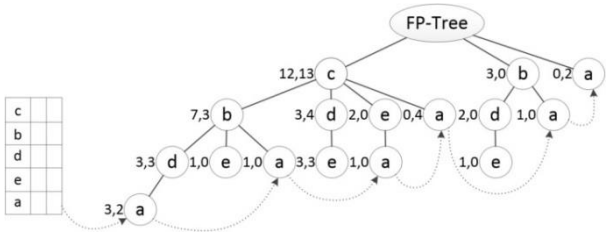
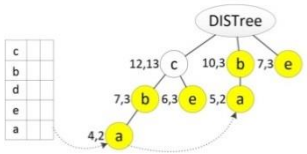


Figure 1: *Header-Table* and *FP-Tree* structure



Itemsets	*Frequency*
{c b a}	*(4,2)*
{c b}	*(7,3)*
{c e}	*(6,3)*
{b a}	*(5,2)*
{b}	*(10,3)*
{e}	*(7,3)*

Figure 2: *DISTree* structure and the discriminative itemsets

Figure 2 also shows the six discriminative itemsets inside the *DISTree* for the datasets in Table 1. Following the generation of all combinations of frequent itemsets in S_i in the *DISTree* structure, traversing and tagging the itemsets as discriminative or non-discriminative, it was proved that the *DISTree* algorithm shows full accuracy and recall in a batch of transactions [2].

5 THE DISSPARSE METHOD

There are two main issues with the *DISTree* method. Firstly, it generates the itemsets if they are frequent in the target dataset without considering their frequency in the general dataset. Secondly, the candidate itemsets remain in the *DISTree* structure while they are checked for being discriminative. The *DISTree* can become extremely massive, making the method unacceptable for large and complex datasets in terms of efficiency.

The key novel idea of the *DISSparse* method is to limit the discriminative itemset mining to the subtrees of the *DISTree* structure which have the potential to contain discriminative itemsets, instead of mining the entire *DISTree* structure. To this end, two heuristics are proposed to determine the potential subtrees and the potential internal nodes in the subtrees for mining discriminative itemsets. The subtrees which do not have potential for containing discriminative itemsets are ignored without checking. The new data structure is defined below.

Conditional FP-Tree: The conditional pattern of a *Header-Table* item is the sub-pattern base under the condition that the item exists in the original *FP-Tree* (e.g., in Figure 1 the conditional patterns of item a are $cbda_{3,2}$, $cba_{1,0}$, $cea_{1,0}$, $ca_{0,4}$, $ba_{1,0}$, $a_{0,2}$). The conditional *FP-Tree* is constructed for each item in the *Header-Table* based on its conditional patterns. In contrast with *FP-Growth* based methods, for the *DISSparse* method, the *Header-Table* item nodes are associated with the top ancestor on the first level of the conditional *FP-Tree* (e.g., Figure 3 is the conditional *FP-Tree* of item a, node c is the top ancestor of the *Header-Table* item a in the left-most subtree, c appears in all *Header-Table* item a's nodes in the left-most subtree). The nodes in the first level of the conditional *FP-Tree* determine different subtrees. Each subtree is made of branches under the same root in the first level of the conditional *FP-Tree* and ending with the processing *Header-Table* item, denoted as $Subtree_{root}$ (e.g., the conditional *FP-Tree* in Figure 3 has three subtrees under root c, b and a i.e., $Subtree_c$, $Subtree_b$ and $Subtree_a$, respectively).

The *Header-Table* item nodes, which are linked using *Header-Table* links, are annotated with the frequency of the itemsets ending with these header items (e.g., $a_{3,2}$ in Figure 3 indicates that the frequency of itemset $cbda$ is 3 and 2 in S_i and S_j respectively, 3 and 2 are the frequency of a with respect to itemset $cbda$). For simplicity, we use a single header item $a_{3,2}$ to denote an itemset $cbda$ and its frequency. Let $Header\_Table\_items(Subtree_{root})$ be a set of header items with their frequency, for each item $a_{n,m} \in Header\_Table\_items(Subtree_{root})$, $I(a_{n,m})$ is defined as the itemset starting from $root$ and ending with a and the frequency of $I(a_{n,m})$ in S_i and S_j are n and m, respectively (e.g., in Figure 3 $Header\_Table\_items(Subtree_c)$ contains $a_{3,2}$, $a_{1,0}$ and $a_{1,4}$, $I(a_{3,2}) = cbda$, $f_i\left(I(a_{3,2})\right) = 3$ and $f_j\left(I(a_{3,2})\right) = 2$).

5.1 Mining Discriminative Itemsets using Sparse Prefix Tree

Following *FP-Growth*, the *FP-Tree* is generated as in Figure 1. The conditional patterns and conditional *FP-Tree* for each *Header-Table* item is then produced following the increasing order of the items' frequency in *Header-Table* starting from the item with the lowest frequency (e.g., the conditional *FP-Tree* in Figure 3 for the *Header-Table* item a which has the lowest frequency).

The conditional *FP-Tree* is traversed from the left-most subtree through processing the *Header-Table* item links. The left-most subtree in conditional *FP-Tree* and its internal nodes are checked based on two heuristics to identify the potential subtrees for mining discriminative itemsets. The two heuristics are defined based on two important measures, the maximum frequency of the itemsets in a subtree and the maximum discriminative value of the itemsets in a subtree which are defined below.

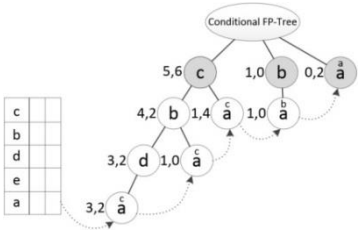


Figure 3: Conditional *FP-Tree* of *Header-Table* item a

Let $itemsets(root, a)$ denote the set of itemsets in subtree $Subtree_{root}$ starting from $root$ and ending with a header item $a_{n,m} \in Header\_Table\_items(Subtree_{root})$. The maximum

frequency of $itemsets(root, a)$ in the target dataset S_i, denoted as $\text{Max\_freq}_i(root, a)$, is defined as the sum of the frequencies in S_i of the itemsets in $itemsets(root, a)$ as below.

$$\text{Max\_freq}_i(root, a) = \sum_{b \in itemsets(root,a)} f_i(b) \quad (3)$$

In the equation above, $f_i(b)$ refers to the frequency in S_i of an itemset b in subtree $Subtree_{root}$ (e.g., in Figure 3 the maximum frequency of itemsets in S_i in the left-most subtree $Subtree_c$, which contains three itemsets ending with $a_{3,2}$, $a_{1,0}$ and $a_{1,4}$, is equal to 5, i.e., $\text{Max\_freq}_i(c, a) = 5$). Let $\mathcal{S}$ be the power set of $itemsets(root, a)$, i.e., $\mathcal{S} = 2^{itemsets(root,a)}$ and $B \in \mathcal{S}$, the discriminative value of the itemsets in B is defined as.

$$\text{Dis\_value}(B) = \begin{cases} \frac{r_i(B)}{\theta} & \sum_{b \in B} f_j(b) = 0 \\ \frac{r_i(B)}{r_j(B)} & \sum_{b \in B} f_j(b) > 0 \end{cases} \quad (4)$$

Where $r_i(B) = \frac{\sum_{b \in B} f_i(b)}{n_i}$ and $r_j(B) = \frac{\sum_{b \in B} f_j(b)}{n_j}$. $r_i(B)$ is called the relative support of B in S_i. The maximum discriminative value of all itemsets in $Subtree_{root}$ ending with a header item a, denoted as $\text{Max\_dis\_value}(root, a)$, is defined as.

$$\text{Max\_dis\_value}(root, a) = max_{B \in \mathcal{S}}\{\text{Dis\_value}(B)\} \quad (5)$$

In order to determine $\text{Max\_dis\_value}(root, a)$, all possible itemsets in $Subtree_{root}$ will have to be generated as required in equations (4) and (5). An efficient method is designed to find a subset B of $itemsets(root, a)$ which makes $\text{Dis\_value}(B)$ maximum among all subsets. The method is omitted due to space limitation. The first heuristic is formally defined below.

HEURISTIC 1. A $Subtree_{root}$ in the conditional *FP-Tree* is considered as a potential discriminative subtree denoted as $Potential(Subtree_{root})$ if it satisfies the following conditions:
(1). $\text{Max\_freq}_i(root, a) \geq \varphi\theta n_i$; (2). $\text{Max\_dis\_value}(root, a) \geq \theta$;

Lemma 1 (Potential discriminative subtree) **HEURISTIC 1** ensures that none of the non-potential discriminative subtrees contains any discriminative itemset.

Proof. The first condition in **HEURISTIC 1** ensures that the sum of frequencies in S_i of itemsets in a potential discriminative subtree is frequent, which implies that there could be an itemset in S_i in the subtree which is frequent. The second condition ensures that the maximum discriminative value of a potential discriminative subtree is larger than the discriminative level θ, which implies that there could be an itemset in the subtree whose discriminative value is larger than the θ.

For a subtree, if it does not satisfy any of the two conditions, the subtree is considered as a non-potential discriminative subtree. Because a non-potential discriminative subtree breaches one or both of the conditions, the subtree does not contain any frequent itemset in S_i or does not contain any itemset whose discriminative value is larger than the discriminative level θ. According to *Definition 1*, the subtree would not contain any discriminative itemset.

■

In Figure 3, the left-most subtree related to the *Header-Table* item a under root node c is potential with $\text{Max\_freq}_i(c, a) = 5 \geq 3$ assuming $\varphi = 0.1, \theta = 2, n_i = 15,$ and $\text{Max\_dis\_value}(c, a) = 2 \geq 2$.

Using **HEURISTIC 1** all potential discriminative subtrees can be identified and all non-potential subtrees are to be ignored from the itemset combination generation. A $Potential(Subtree_{root})$ could contain non-discriminative itemsets. The non-potential subsets may exist in $Potential(Subtree_{root})$ as internal nodes which are in the paths between $root$ of $Subtree_{root}$ and the items in $Header\_Table\_items(Subtree_{root})$ denoted as $Internal\ node_{root}$ (e.g., in the left-most subtree in conditional *FP-Tree* in Figure 3, $Potential(Subtree_c)$, has two internal nodes i.e., b_c and d_c, respectively).

Let $itemsets(root, in, a)$ denote a set of itemsets in subtree $Subtree_{root}$ ending with a header item $a \in Header\_Table\_items(Subtree_{root})$ with the internal node in as subset i.e., $itemsets(root, in, a) \subseteq itemsets(root, a)$. The maximum frequency of $itemsets(root, in, a)$ in the target dataset S_i is denoted as $\text{Max\_freq}_i(root, in, a)$. The maximum discriminative value of $itemsets(root, in, a)$ is denoted as $\text{Max\_dis\_value}(root, in, a)$. The second heuristic is formally defined below.

HEURISTIC 2. An internal node $in \in Internal\ node_{root}$ in a $Potential(Subtree_{root})$ is considered as potential discriminative internal node denoted as $Potential(in)$ if it satisfies the following conditions:
(1). $\text{Max\_freq}_i(root, in, a) \geq \varphi\theta n_i$; (2). $\text{Max\_dis\_value}(root, in, a) \geq \theta$;

Lemma 2 (Potential discriminative internal node) **HEURISTIC 2** ensures that none of the non-potential discriminative internal nodes would occur in any discriminative itemset.

Proof. The proof is similar to **Lemma 1**, omitted due to space limitation.

■

In Figure 3, the internal node d_c in $Potential(Subtree_c)$ is non-potential as $\text{Max\_freq}_i(c, d, a) = 3$ and $\text{Max\_dis\_value}(c, d, a) = 1.5$. The non-potential internal nodes are ignored from itemset combination generation in $Potential(Subtree_{root})$.

5.1.1 Potential Discriminative Itemsets Generation using Minimized DISTree

The *DISSparse* method does not use the *DISTree* structure to identify discriminative itemsets, which is usually very big. Instead, it identifies potential discriminative subtrees in the conditional *FP-Tree*, and then discovers discriminative itemsets from the potential discriminative subtrees, which can significantly increase the efficiency. The potential discriminative

itemsets identified from a $Subtree_{root}$ in conditional *FP-Tree* are discovered in a minimized *DISTree* structure defined below.

Minimized DISTree: The minimized *DISTree* is similar to the *DISTree* structure defined early in previous section [2]. The size of a minimized *DISTree* is bounded to the size of potential subsets in one potential discriminative subtree in a conditional *FP-Tree* and non-potential subsets are ignored when the itemset combinations are generated (e.g., the minimized *DISTree* in Figure 4 is generated out of the potential discriminative subsets of the left-most subtree in the conditional *FP-Tree* in Figure 3 without considering d_c which is a non-discriminative internal node). The minimized *DISTree* covers the itemsets starting with $root$ item of $Subtree_{root}$ as prefix and items in $Header\_Table\_items(Subtree_{root})$ as postfix (e.g., the minimized *DISTree* in Figure 4 covers the itemsets with prefix of c and postfix of a, generated out of the potential discriminative subset of items c, b and a in the left-most subtree in conditional *FP-Tree* in Figure 3). This is formally defined below.

Let I be an itemset in a minimized *DISTree* for a given potential discriminative subtree, i.e., $Subtree_{root}$. For all internal subsets I' of itemset I, i.e., $I' \subset I$, which start immediately after the $root$ item of $Subtree_{root}$ and end before the items in $Header\_Table\_items(Subtree_{root})$, the subsets I' are included in the minimized *DISTree* generated from the $Subtree_{root}$ if $I = \{root\} \cup I' \cup \{a\}$, where $a \in Header\_Table\_items(Subtree_{root})$.

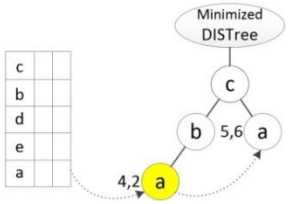


Figure 4: Minimized *DISTree* generated from the left-most subtree in Figure 3

By generating the potential discriminative itemset combinations out of all branches in a potential discriminative subtree (e.g., three branches in the left-most subtree, $Potential(Subtree_c)$ in Figure 3 i.e., $cbda_{3,2}$, $cba_{1,0}$ and $ca_{1,4}$), the minimized *DISTree* is traversed through *Header-Table* item links for mining the discriminative itemsets based on *Definition 1* (e.g., the highlighted node $a_{4,2}$ in Figure 4 refers to the discriminative itemset $cba_{4,2}$).

5.1.2 Conditional FP-Tree Expansion

In a conditional *FP-Tree*, except for the left-most subtree such as the subtree with root c in Figure 3, a subtree with a particular root item may not contain all the itemsets starting with that particular item (e.g., in Figure 3, the itemset bda, which was included in the left-most subtree, was not included in the subtree with root b). In order to generate all the possible discriminative itemsets, before traversing the next $Subtree_{root}$ of the processing *Header-Table* item, the conditional *FP-Tree* must be expanded by adding the sub-branches of the current $Subtree_{root}$ without their $root$ item. Each sub-branch is added to the conditional *FP-Tree* under the $root$ node of one of the remaining subtrees and the frequencies of the itemsets in the remaining subtrees are updated by summing up the frequencies of the itemsets ending with the items in $Header\_Table\_items(Subtree_{root})$ (e.g., three sub-branches in the left-most subtree, $Subtree_c$ in Figure 3 i.e., $bda_{3,2}$, $ba_{1,0}$ and $a_{1,4}$ are added to the conditional *FP-Tree* as in Figure 5 to generate three branches under root b or a i.e., $bda_{3,2}$, $ba_{2,0}$ and $a_{1,6}$). For space saving, the $Header\_Table\_items(Subtree_{root})$ of the processed $Subtree_{root}$ are removed from conditional *FP-Tree*. The conditional *FP-Tree* expansion continues by processing each $Subtree_{root}$ until no further $Subtree_{root}$ remains.

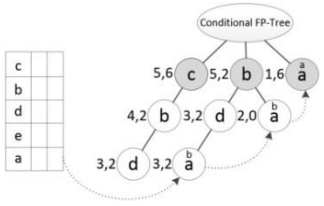


Figure 5: Expanded conditional *FP-Tree* of *Header-Table* item a after processing the first subtree

In Example 1, the $Subtree_b$ is traversed by $Header\_Table\_items(Subtree_b)$ links. The $Subtree_b$ has two branches ending with the $Header\_Table\_items(Subtree_b)$ i.e., $I(a_{3,2})$ and $I(a_{2,0})$ as in Figure 5. Based on **HEURISTIC 1** the $Subtree_b$ is potential with $\text{Max\_freq}_i(b,a) = 5$ and $\text{Max\_dis\_value}(b,a) = 2.5$. Based on **HEURISTIC 2** the internal node d is non-potential with $\text{Max\_freq}_i(b,d,a) = 3$ and $\text{Max\_dis\_value}(b,d,a) = 1.5$. The minimized *DISTree* for the *Header-Table* item a based on potential discriminative subsets in the left-most subtree in conditional *FP-Tree* in Figure 5, $Subtree_b$, is generated as in Figure 6 (e.g., the highlighted node $a_{5,2}$ in Figure 6 refers to the discriminative itemset $ba_{5,2}$).

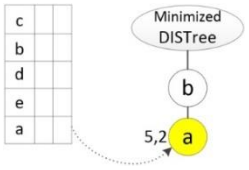


Figure 6: Minimized *DISTree* generated from the left-most subtree in Figure 5

The conditional *FP-Tree* of the *Header-Table* item a is then expanded by adding the two sub-branches of $Subtree_b$, $da_{3,2}$ and $a_{2,0}$ as in Figure 7. Based on **HEURISTIC 1** the $Subtree_d$ with one itemset ending the items in $Header\_Table\_items(Subtree_d)$ i.e., $I(a_{3,2})$ is non-potential with $\text{Max\_freq}_i(d,a) = 3$ and $\text{Max\_dis\_value}(d,a) = 1.5$, and the minimized *DISTree* is not generated. Based on the completeness of itemset prefixes, the conditional *FP-Tree* is

expanded by adding the sub-branches of the non-potential $Subtree_{root}$ as well (e.g., the conditional *FP-Tree* in Figure 7 is expanded by adding the single sub-branch of $Subtree_d$, $a_{3,2}$). The $Subtree_a$ with one itemset (i.e., $I(a_{6,8})$) is non-potential.

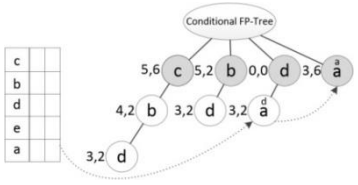


Figure 7: Expanded conditional *FP-Tree* of *Header-Table* item a after processing the second subtree

The potential discriminative itemset combination generation for the processing *Header-Table* item (i.e., item a in Figure 7) is finished if there is no more $Subtree_{root}$ in the conditional *FP-Tree*. Then, the next least frequent item (i.e., item e in Example 1), is processed to generate the conditional *FP-Tree*, and the process continues for the rest of *Header-Table* items respectively (i.e., items d, b, c in Example 1), and by processing the last *Header-Table* item (i.e., item c in Example 1) all the discriminative itemsets are reported as in DI_{ij}.

5.2 DISSparse Algorithm

The *DISSparse* algorithm starts by reading the batch of transactions, and making the *FP-Tree* and *Header-Table* similar to *FP-Growth* [14]. Starting from the least frequent item, a conditional *FP-Tree* is built for each *Header-Table* item. Every $Subtree_{root}$ in the conditional *FP-Tree* is assessed using **HEURISTIC 1** and **HEURISTIC 2** to limit the algorithm to the potential discriminative itemsets. The minimized *DISTree* is generated for a $Potential(Subtree_{root})$ by the potential discriminative itemset combinations and discriminative itemsets are reported in DI_{ij}. The conditional *FP-Tree* is expanded by the sub-branches of the processed $Subtree_{root}$ and the process continues if there are remaining subtrees.

In the *DISSparse* algorithm, the parts attracting most of the complexity include the finding of potential discriminative itemsets, generating the minimized *DISTree* and conditional *FP-Tree* expansion. The minimized *DISTree* is a concise data structure and it is generated only from potential discriminative subsets, therefore, the number of generated potential discriminative itemsets in a minimized *DISTree* is significantly fewer than the exponential number of frequent itemsets as generated in *DISTree* method. Therefore, compared to the *DISTree* method, the discriminative itemsets can be instantly discovered from the minimized *DISTree*. At any time there is only one conditional *FP-Tree* which is a small data structure. The expansion with the sub-branches of the current subtree is linear.

Based on the two lemmas and the conditional *FP-Tree* expansion, we can prove that the *DISSparse* algorithm can find all correct discriminative itemsets. The proof is omitted due to space limitation.

Algorithm 1 *DISSparse*
Input: (1) Discriminative level threshold θ; (2) Support threshold φ; (3) Batch of transactions B of datasets S_i and S_j.
Output: DI_{ij}, discriminative itemsets in S_i against S_j.
Begin
1. Scan B to order the items in transactions by frequency;
2. Make *FP-Tree* and *Header-Table* for B;
3. DI_{ij} ={ };
4. **For** each item *x* in *Header-Table* **do** // *x is least-frequent*
5. Make conditional *FP-Tree*$_x$ based on item *x;*
6. **While** conditional *FP-Tree*$_x$ has remaining $Subtree_{root}$ **do** // *left-most order*
7. Assess $Subtree_{root}$ using **HEURISTIC 1**;
8. **If** $Potential(Subtree_{root})$ **then**
9. Find $Potential(Internal\ node_{root})$ using **HEURISTIC 2**;
10. Generate minimized *DISTree* based on $Potential(Subtree_{root})$ with only potential internal nodes in $Potential(Internal\ node_{root})$ and update DI_{ij} by discovered discriminative itemsets;
11. **End if;**
12. Expand conditional *FP-Tree*$_x$ by sub-branches of $Subtree_{root}$;
13. **End while;**
14. **End for;**
15. Report discriminative itemsets in DI_{ij};
End.

6 PERFORMANCE EVALUATION

The algorithms were implemented in C++ and the experiments were conducted on a desktop computer with an Intel Core (TM) Duo E2640 2.8GHz CPU and 8GB main memory running 64 bit Microsoft Windows 7 Enterprise. The synthetic datasets were generated using the IBM synthetic data generator [15]. The $T\$: I\$: D\$$ format shows the datasets with T as the average transaction length, I as the average length of the maximal potentially large itemsets and D as the number of transactions. We used the same T for S_1 and S_2 to indicate that both datasets belong to the same domain. Both S_1 and S_2 were generated with different I as there are more maximal potentially large itemsets in S_2 which is made of several smaller datasets. This will also ensure there are a large number of discriminative itemsets in S_1 against S_2. For simplicity, we defined the combination of $\varphi\theta n_1$ as minimum support.

Although the *DPMiner* algorithm [3] is to mine discriminative itemsets with $f_i > minimum\ support$ i.e., itemset must be frequent in the target dataset, but it has a specific requirement on $f_j < \delta$, which is different from the measure used by *DISSparse* to determine discriminative itemsets. Therefore, in the evaluation below, the *DISTree* algorithm is chosen as a baseline model to compare with the proposed *DISSparse* algorithm.

6.1 Evaluation on Time and Space Efficiency

The *DISTree* and *DISSparse* algorithms are evaluated in three synthetic datasets, each of which made of target dataset S_1 and general dataset S_2. The S_2 is typically much bigger than S_1 as it combines multiple datasets. The scalability of *DISTree* and

DISSparse algorithms is presented within different ranges of discriminative level θ, support threshold φ, ratios between S_1 and S_2 (i.e., n_2/n_1) and also the number of unique items in $\sum$. The experiment with real dataset is omitted due to space limitation.

Dataset D_1 is generated with S_1 as $T25{:}I10{:}D10K$ and S_2 as $T25{:}I15{:}D50K$ limited to $1K$ unique items. The scalability of *DISSparse* is compared with *DISTree*, by different θ and a fixed $\varphi = 0.01\%$, as presented in Figure 8.

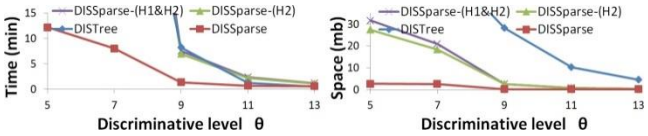


Figure 8: Scalability with θ for D_1 ($\varphi = 0.0001$)

DISSparse scales much better than *DISTree* specifically for lower minimum supports when a greater number of discriminative itemsets are discovered. The algorithms have close behavior in large minimum supports when there is less number of discriminative itemsets (i.e., for $\theta = 11 - 13$ the number of discriminative itemsets is less than 150k). By large minimum supports both *DISTree* and *DISSparse* prune many items that are infrequent in the target dataset S_1 during the making of conditional *FP-Tree*. In Figure 8 the exponential growth in time and space usage of *DISTree* with smaller minimum support is observed caused by exponential number of generated itemset combinations.

For showing the efficiency gained by the proposed heuristics, the experiments are repeated by testing the *DISSparse* algorithm without using **HEURISTIC 2** (denoted as *DISSparse* - (H2)) and without using both heuristics (denoted as *DISSparse* - (H1&H2)). The proposed *DISSparse* algorithm without using the proposed heuristics is not efficient and *DISSparse* does not scale well even compared to *DISTree* for larger θ (e.g., the time complexity of *DISSparse* in Figure 8 without considering the heuristic generally scales higher than *DISTree*). The space usage of *DISSparse* without using the heuristics is still much better compared to *DISTree*. The main contribution regarding the *DISSparse* efficiency is related to the **HEURISTIC 2**, however, the effect of **HEURISTIC 1** becomes clear when the lower minimum supports are considered.

The scalability of the algorithms is tested with different φ and fixed $\theta = 10$ as presented in Figure 9. The φ has similar effect as θ on the time and space complexity as these two parameters together define the minimum support of discriminative itemsets.

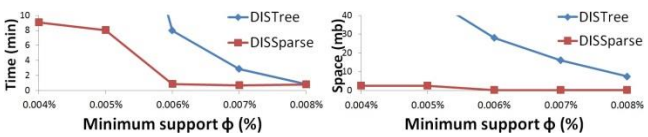


Figure 9: Scalability with φ for D_1 ($\theta = 10$)

The comparison result with different dataset length ratios by changing the number of transactions in S_2 from $10k$ to $100k$ is shown in Figure 10. The runtime in *DISSparse* is mainly increased as a result of rise in the number of discriminative itemsets (i.e., slightly from $185k$ to $195k$). The linear increase in the time and space complexity of *DISTree* by larger ratios is caused by the bigger data structures (i.e., the dataset S_2 with $\frac{n_2}{n_1} = 10$ becomes ten times bigger than with $n_1 = n_2$ when n_1 keeps the same).

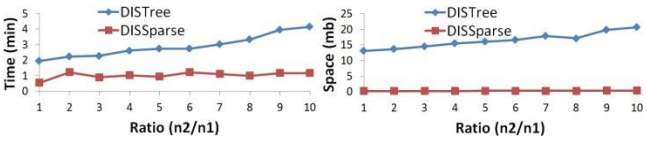


Figure 10: Scalability with different dataset length ratios (n_2/n_1) ($\theta = 10$ and $\varphi = 0.0001$)

Dataset D_2 is generated with S_1 as $T25{:}I10{:}D10K$ and S_2 as $T25{:}I15{:}D50K$ limited to $10K$ unique items, 10 times more than the items in D_1, which makes items less frequent than in D_1. The number and average length of discriminative itemsets decrease. Both algorithms prune many itemsets that are infrequent in S_1 by making conditional *FP-Tree*. *DISSparse* scales better than *DISTree* for both time and space even *DISTree* scales with a close time complexity to the *DISSparse* for larger θ but larger space usage as in Figure 11.

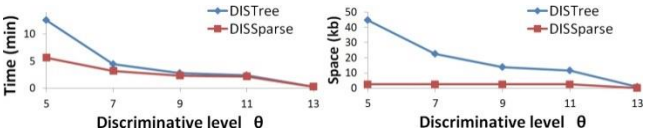


Figure 11: Scalability with θ for D_2 ($\varphi = 0.0001$)

Dataset D_3 is generated with S_1 as $T25{:}I10{:}D50K$ and S_2 as $T25{:}I15{:}D250K$ limited to $10K$ unique items. The n_1 is much bigger in D_3 than n_1 in D_2 and D_1 that caused larger minimum support and consequently less number of discriminative itemsets (e.g., for minimum support $\varphi\theta n_1 = 25..95$ the number of discriminative itemsets with different θ varies from 3 million to a few hundred). Both algorithms prune many infrequent itemsets in S_1 by making conditional *FP-Tree* and show close performances together as in Figure 12. The *FP-Tree* construction pushes an overhead of a minute to the processing time of both methods with considerable space usage.

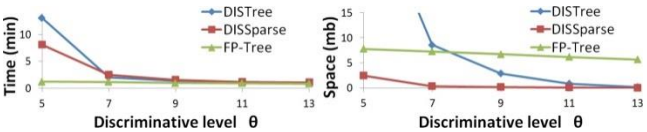


Figure 12: Scalability with θ for D_3 ($\varphi = 0.0001$)

DISSparse scales very well for large datasets by a prolific number of discriminative itemsets with smaller minimum supports as in Figure 13. The experiments are conducted on D_3

using smaller support threshold (i.e., the similar parameter setting with experiments on D_1 reported in Figure 8). The support threshold is set to $\varphi = 0.00002$ for having minimum support, $\varphi\theta n_1 = 0.00002 * \theta * 50{,}000 = \theta$. In small minimum supports the *DISTree* becomes intolerable by an exponential number of discriminative itemsets (i.e., from $500K$ for $\theta = 35$ to $9M$ for $\theta = 15$).

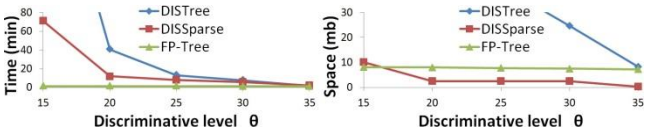


Figure 13: Scalability with θ for D_3 ($\varphi = 0.00002$)

7 CONCLUSIONS

This paper proposed an efficient method for mining discriminative itemsets based on satisfactory heuristics. The proposed method has been extensively evaluated with datasets exhibiting distinct characteristics. Empirical analysis shows the proposed heuristics resulted in significant improvement in time and memory consumption. The algorithm reports the discriminative itemsets with full accuracy and recall. In this paper, the multiple datasets treated as by combining them as a general dataset. The method remains the same principally if the multiple datasets need to be considered separately.

REFERENCES

[1] Z. Lin, B. Jiang, J. Pei, and D. Jiang, "Mining discriminative items in multiple data streams," *World Wide Web*, vol. 13, pp. 497-522, 2010.

[2] M. Seyfi, S. Geva, and R. Nayak, "Mining Discriminative Itemsets in Data Streams," in *International Conference on Web Information Systems Engineering*, 2014, pp. 125-134.

[3] G. Dong and J. Bailey, *Contrast Data Mining: Concepts, Algorithms, and Applications*: CRC Press, 2012.

[4] G. Dong and J. Li, "Efficient Mining of Emerging Patterns: Discovering Trends and Differences," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 43-52.

[5] J. Li, G. Liu, and L. Wong, "Mining statistically important equivalence classes and delta-discriminative emerging patterns," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 430-439.

[6] X. Zhang, G. Dong, and R. Kotagiri, "Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 310-314.

[7] H. Fan and K. Ramamohanarao, "An Efficient Single-Scan Algorithm for Mining Essential Jumping Emerging Patterns for Classification," in *Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, 2002, pp. 456-462.

[8] J. Bailey, T. Manoukian, and K. Ramamohanarao, "Fast Algorithms for Mining Emerging Patterns," in *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, 2002, pp. 39-50.

[9] H. Alhammady and K. Ramamohanarao, "Mining Emerging Patterns and Classification in Data Streams," *The Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 272-275 2005.

[10] J. Bailey and E. Loekito, "Efficient incremental mining of contrast patterns in changing data," *Information processing letters*, vol. 110, pp. 88-92, 2010.

[11] K. Yu, W. Ding, H. Wang, and X. Wu, "Bridging causal relevance and pattern discriminability: Mining emerging patterns from high-dimensional data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 2721-2739, 2013.

[12] K. Yu, W. Ding, D. A. Simovici, H. Wang, J. Pei, and X. Wu, "Classification with Streaming Features: An Emerging-Pattern Mining Approach," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, pp. 1-31, 2015.

[13] M. Seyfi, "Mining discriminative items in multiple data streams with hierarchical counters approach," in *Fourth International Workshop on Advanced Computational Intelligence (IWACI)*, 2011, pp. 172-176.

[14] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM Sigmod Record*, 2000, pp. 1-12.

[15] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases VLDB*, 1994, pp. 487-499.